Question:4 (All coding is in python. Please find it in attachment. File name=Greedy_Naive.py)

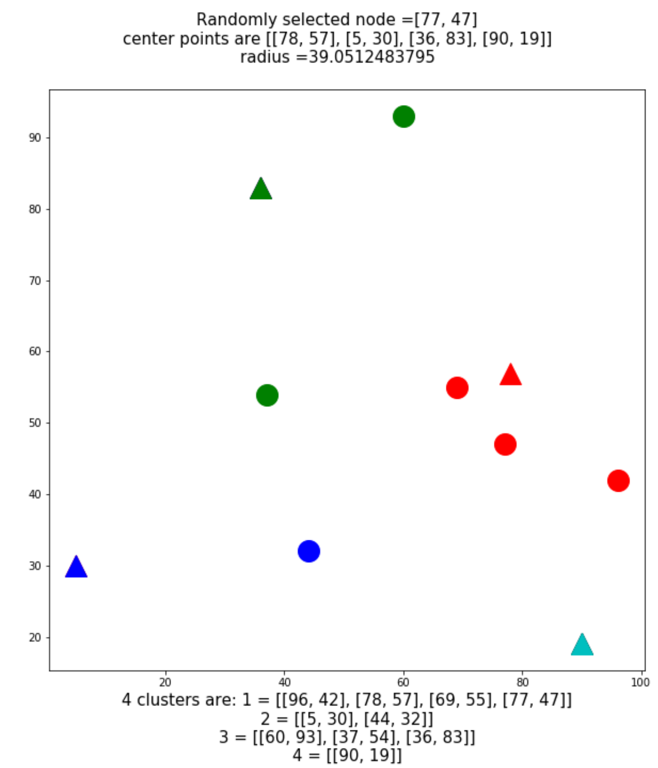
(60%) In the algorithm for the k-center problem, the first cluster head is chosen randomly. A naïve improvement is to try every node as the first cluster head and then find the corresponding partition. Therefore, we obtain n = |V| partitions. Finally, we simply output the best partition among these n partitions, i.e., the one that has the smallest maximum radius. Write a program to implement and compare the original greedy algorithm and the above naïve improvement. In your report, you need to explain how you construct the problem instances.

Solution:

The algorithm which I designed is completely dynamic. It works for all values of k and any number of nodes. The colors represent different clusters, the triangle represent cluster head and the circles represent cluster members.

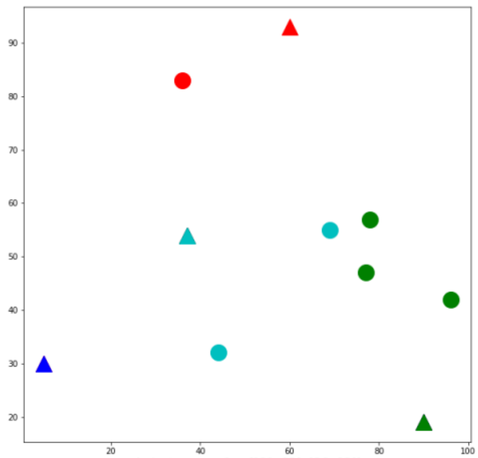
Example1: As a first example I will take k=4 and "number of vertices" = 10
The vertices are filled with random numbers between 0 and 100.
vertices = [[60, 93], [96, 42], [37, 54], [90, 19], [5, 30], [78, 57], [36, 83], [69, 55], [44, 32], [77, 47]]

Greedy search algorithm: The algorithm selected node [77, 47] and radius was 39.05. The following figure shows the 4 clusters (graphical as well as quantitative data).



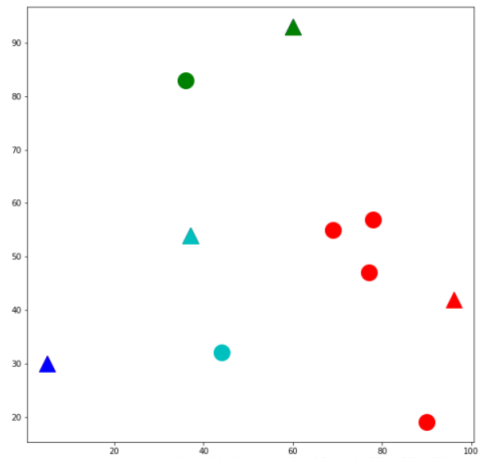
Naïve Improvement: After this the algorithm moves to find the naïve improvement. We start from node 1 and iterate through all the nodes. Instead of random selection enforce each node to be the first in every iteration.

Forced selection of starting node 1=[60, 93]
center points are [[60, 93], [5, 30], [90, 19], [37, 54]]radius =39.8497176903



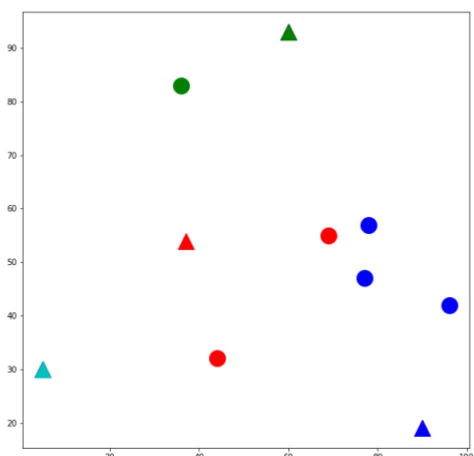
4 clusters are: 1 = [[60, 93], [36, 83]]
2 = [[5, 30]]
3 = [[96, 42], [90, 19], [78, 57], [77, 47]]
4 = [[37, 54], [69, 55], [44, 32]]

Forced selection of starting node 2=[96, 42]
center points are [[96, 42], [5, 30], [60, 93], [37, 54]]radius =29.9666481275



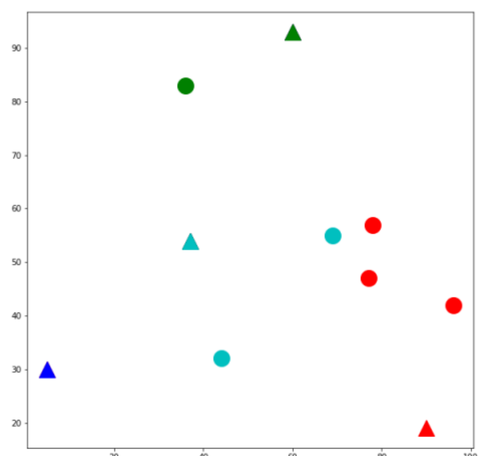
4 clusters are: 1 = [[96, 42], [90, 19], [78, 57], [69, 55], [77, 47]]
2 = [[5, 30]]
3 = [[60, 93], [36, 83]]
4 = [[37, 54], [44, 32]]

Forced selection of starting node 3=[37, 54]
center points are [[37, 54], [90, 19], [60, 93], [5, 30]]radius =39.8497176903



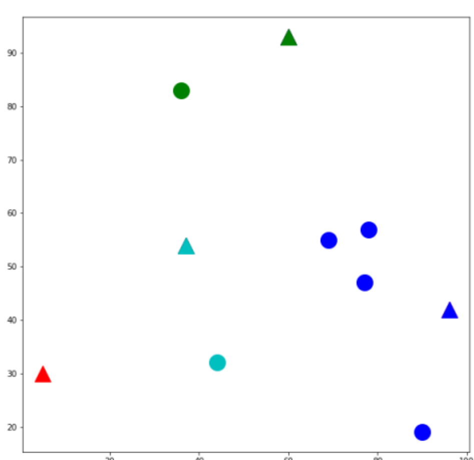
4 clusters are: 1 = [[37, 54], [69, 55], [44, 32]]
2 = [[96, 42], [90, 19], [78, 57], [77, 47]]
3 = [[60, 93], [36, 83]]
4 = [[5, 30]]

Forced selection of starting node 4=[90, 19]
center points are [[90, 19], [5, 30], [60, 93], [37, 54]]radius =39.8497176903



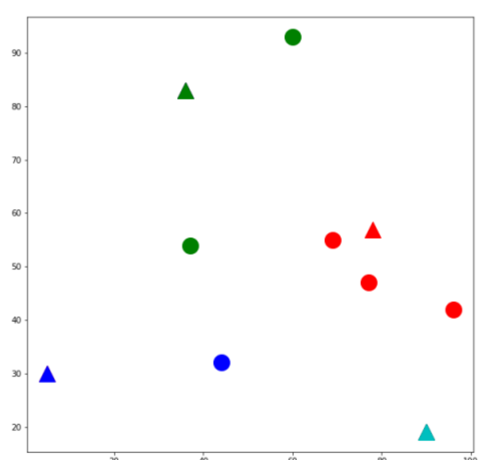
4 clusters are: 1 = [[96, 42], [90, 19], [78, 57], [77, 47]]
2 = [[5, 30]]
3 = [[60, 93], [36, 83]]
4 = [[37, 54], [69, 55], [44, 32]]

Forced selection of starting node 5=[5, 30]
center points are [[5, 30], [96, 42], [60, 93], [37, 54]]radius =29.9666481275



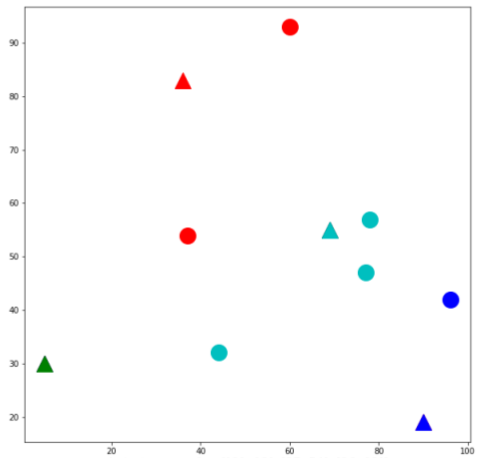
4 clusters are: 1 = [[5, 30]]
2 = [[96, 42], [90, 19], [78, 57], [69, 55], [77, 47]]
3 = [[60, 93], [36, 83]]
4 = [[37, 54], [44, 32]]

Forced selection of starting node 6=[78, 57]
center points are [[78, 57], [5, 30], [36, 83], [90, 19]]radius =39.0512483795



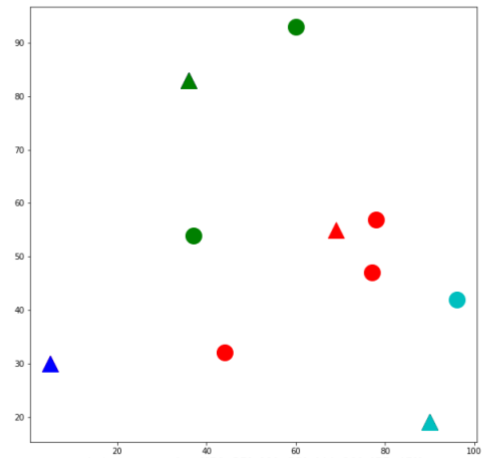
4 clusters are: 1 = [[96, 42], [78, 57], [69, 55], [77, 47]]
2 = [[5, 30], [44, 32]]
3 = [[60, 93], [37, 54], [36, 83]]
4 = [[90, 19]]

Forced selection of starting node 7=[36, 83]
center points are [[36, 83], [90, 19], [5, 30], [69, 55]]radius =33.9705755029



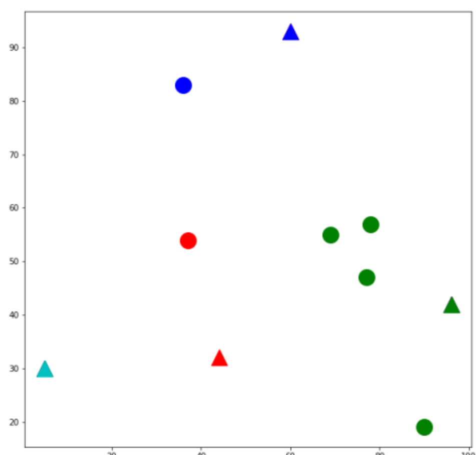
4 clusters are: 1 = [[60, 93], [37, 54], [36, 83]]
2 = [[96, 42], [90, 19]]
3 = [[5, 30]]
4 = [[78, 57], [69, 55], [44, 32], [77, 47]]

Forced selection of starting node 8=[69, 55]
center points are [[69, 55], [5, 30], [36, 83], [90, 19]]radius =33.9705755029



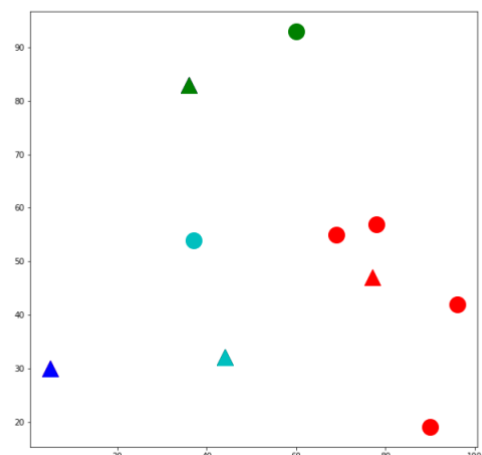
4 clusters are: 1 = [[78, 57], [69, 55], [44, 32], [77, 47]]
2 = [[5, 30]]
3 = [[60, 93], [37, 54], [36, 83]]
4 = [[96, 42], [90, 19]]

Forced selection of starting node 9=[44, 32]
center points are [[44, 32], [60, 93], [96, 42], [5, 30]]radius =29.9666481275



4 clusters are: 1 = [[37, 54], [44, 32]]
2 = [[60, 93], [36, 83]]
3 = [[96, 42], [90, 19], [78, 57], [69, 55], [77, 47]]
4 = [[5, 30]]

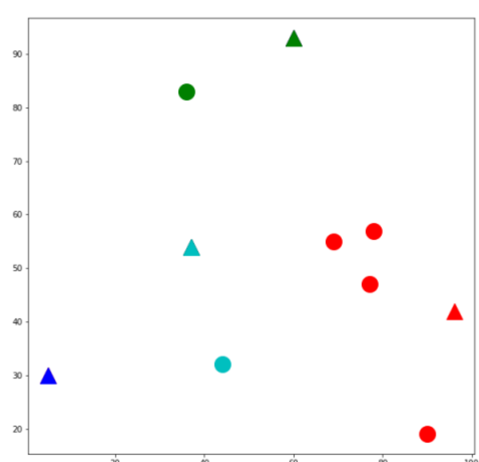
Forced selection of starting node 10=[77, 47]
center points are [[77, 47], [5, 30], [36, 83], [44, 32]]radius =30.8706980809



4 clusters are: 1 = [[96, 42], [90, 19], [78, 57], [69, 55], [77, 47]]
2 = [[5, 30]]
3 = [[60, 93], [36, 83]]
4 = [[37, 54], [44, 32]]

Best selection: The algorithm selects the best of these on the basis of minimum radius. The best of all these forced selections is [96, 42] whose radius is 29.96.

Forced selection of starting node 2=[96, 42]
center points are [[96, 42], [5, 30], [60, 93], [37, 54]]radius =29.9666481275

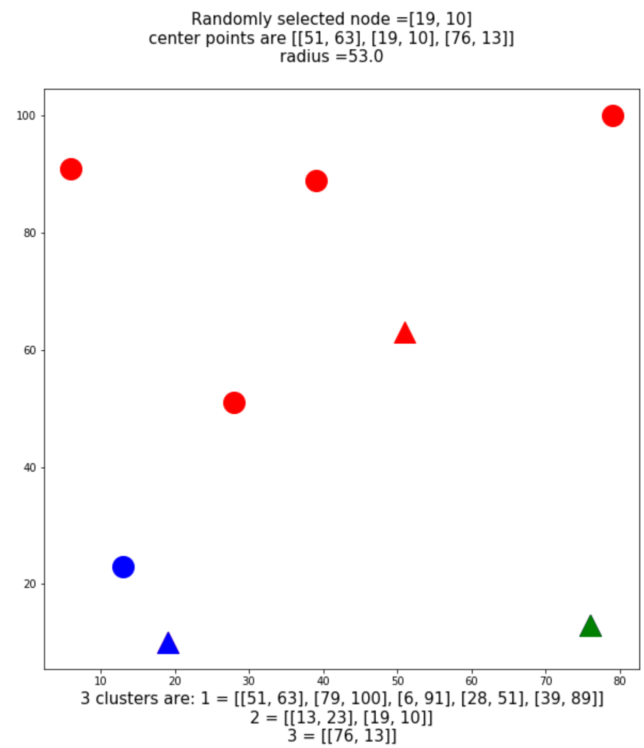


4 clusters are: 1 = [[96, 42], [90, 19], [78, 57], [69, 55], [77, 47]]
2 = [[5, 30]]
3 = [[60, 93], [36, 83]]
4 = [[37, 54], [44, 32]]

Comparison of greedy and naïve improvement:
The naïve improvement algorithm selected node [96, 42], and made clusters from which we found that radius is 29.9666481275
The Greedy search algorithm selected node [78, 57] and made clusters from which we found that radius is 39.0512483795
Hence its proved that selection of best choice will always be better. The radius of naïve improvement will always make clusters in a way that radius will be smaller or equal to greedy search algorithm and will never be greater.

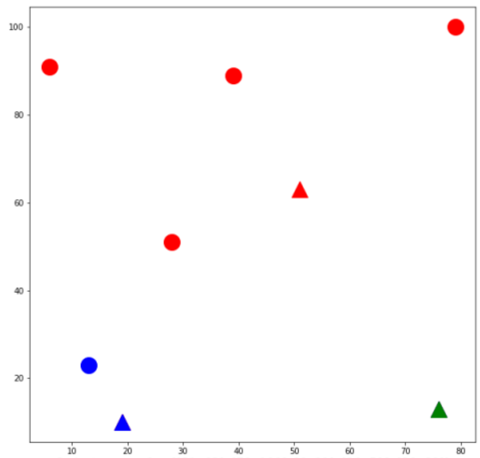
Example 2: In this example I will take k=3 and “number of vertices” = 8.
The vertices are filled with random numbers between 0 and 100.
vertices = [[51, 63], [79, 100], [6, 91], [76, 13], [13, 23], [28, 51], [39, 89], [19, 10]]

Greedy search algorithm was first applied. The algorithm selected node [19, 10] and radius was 53.0.
The following figure shows the 3 clusters (graphical as well as the quantitative data).



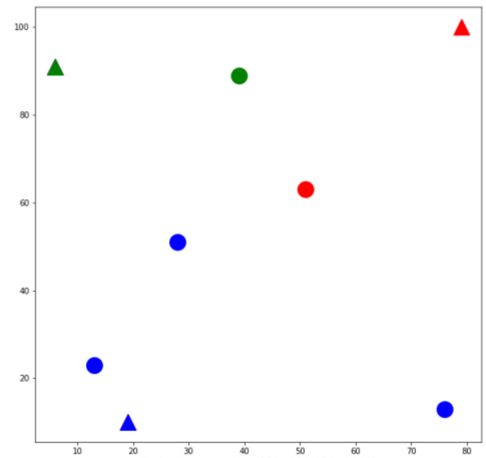
Naïve Improvement: After this the algorithm moves to find the naïve improvement. We start from node 1 and iterate through all the nodes to find the best one.

Forced selection of starting node 1=[51, 63]
center points are [[51, 63], [19, 10], [76, 13]]radius =53.0



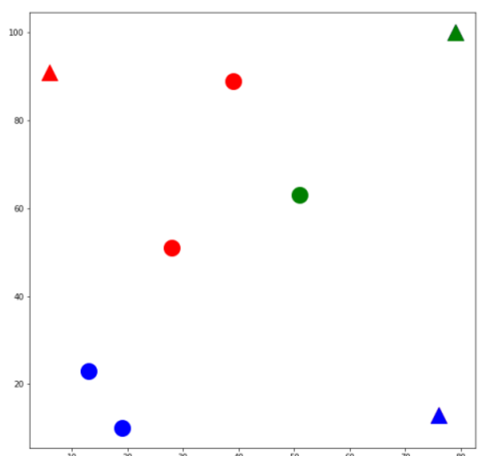
3 clusters are: 1 = [[51, 63], [79, 100], [6, 91], [28, 51], [39, 89]]
2 = [[13, 23], [19, 10]]
3 = [[76, 13]]

Forced selection of starting node 2=[79, 100]
center points are [[79, 100], [19, 10], [6, 91]]radius =57.0788927713



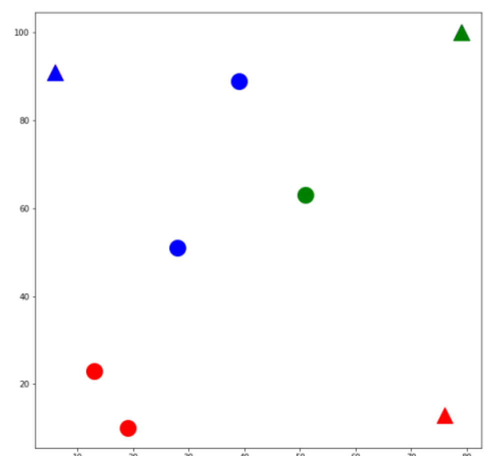
3 clusters are: 1 = [[51, 63], [79, 100]]
2 = [[76, 13], [13, 23], [28, 51], [19, 10]]
3 = [[6, 91], [39, 89]]

Forced selection of starting node 3=[6, 91]
center points are [[6, 91], [76, 13], [79, 100]]radius =63.7887137353



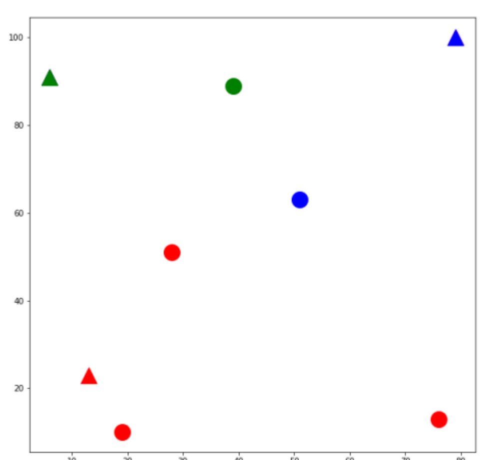
3 clusters are: 1 = [[6, 91], [28, 51], [39, 89]]
2 = [[76, 13], [13, 23], [19, 10]]
3 = [[51, 63], [79, 100]]

Forced selection of starting node 4=[76, 13]
center points are [[76, 13], [6, 91], [79, 100]]radius =63.7887137353



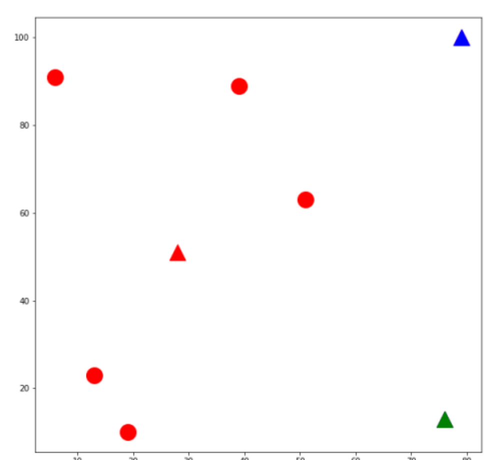
3 clusters are: 1 = [[76, 13], [13, 23], [19, 10]]
2 = [[6, 91], [28, 51], [39, 89]]
3 = [[51, 63], [79, 100]]

Forced selection of starting node 5=[13, 23]
center points are [[13, 23], [79, 100], [6, 91]]radius =63.7887137353

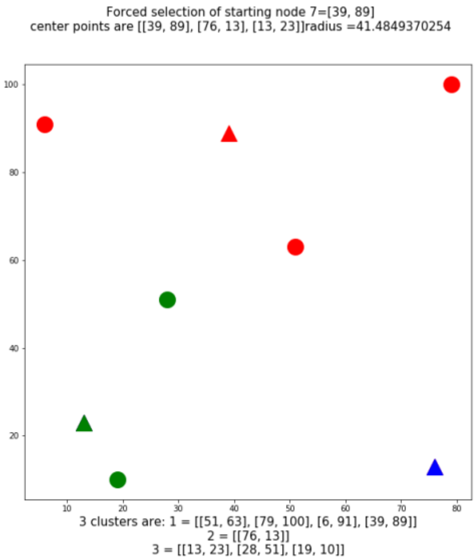


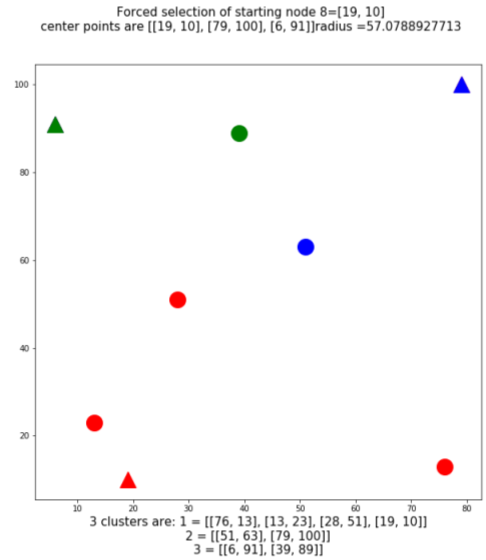
3 clusters are: 1 = [[76, 13], [13, 23], [28, 51], [19, 10]]
2 = [[51, 63], [79, 100]]
3 = [[6, 91], [39, 89]]

Forced selection of starting node 6=[28, 51]
center points are [[28, 51], [79, 100], [76, 13]]radius =45.6508488421

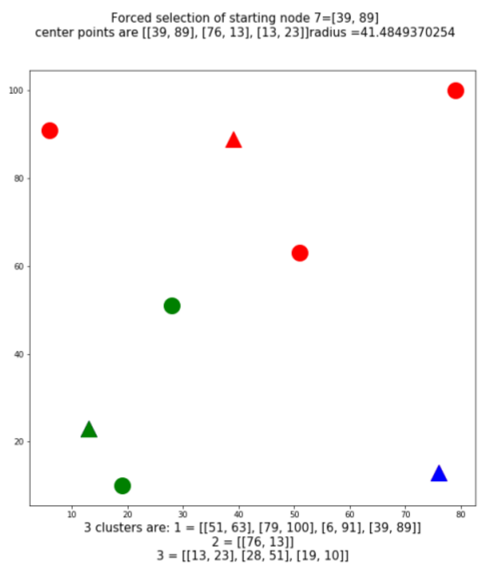


3 clusters are: 1 = [[51, 63], [6, 91], [13, 23], [28, 51], [39, 89], [19, 10]]
2 = [[79, 100]]
3 = [[76, 13]]





Best selection: The algorithm selects the best of these on the basis of minimum radius. The best of all these forced selections is [39, 89] whose radius is 41.48.



Comparison of greedy and naïve improvement:
The Greedy search algorithm selected node [19, 10], and made clusters from which we found that radius is 53.0
The Naïve improvement selected node [39, 89] and made clusters from which we found that radius is 41.48
So again Naïve improvement gave us better result than greedy.

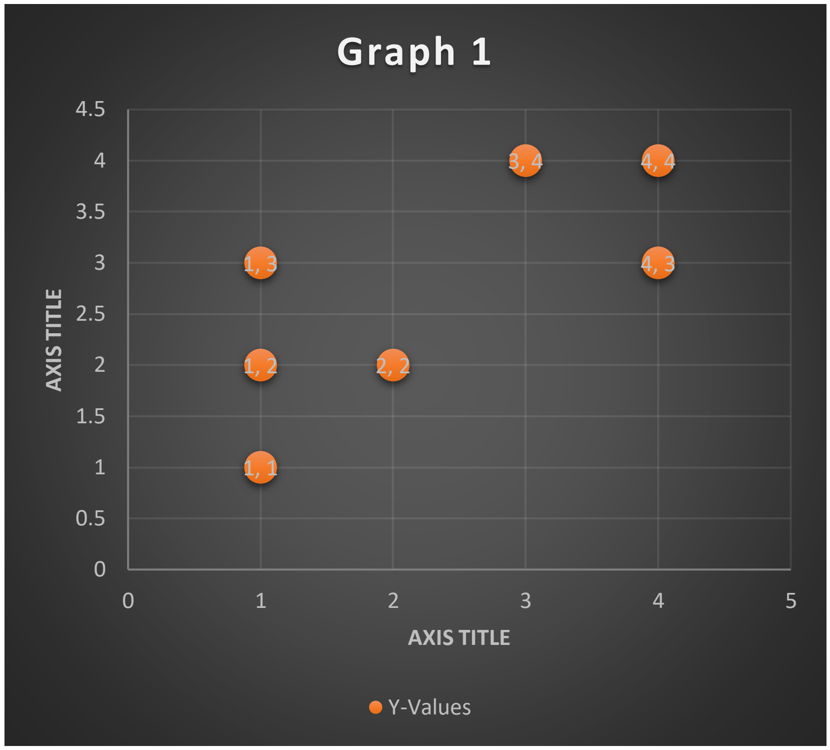
Question 3:

(10%) Consider the k-center problem. Please give a problem instance in which rr∗= 2, where r and r∗ are the maximum radii in the solution obtained by the greedy algorithm and the optimal solution, respectively.

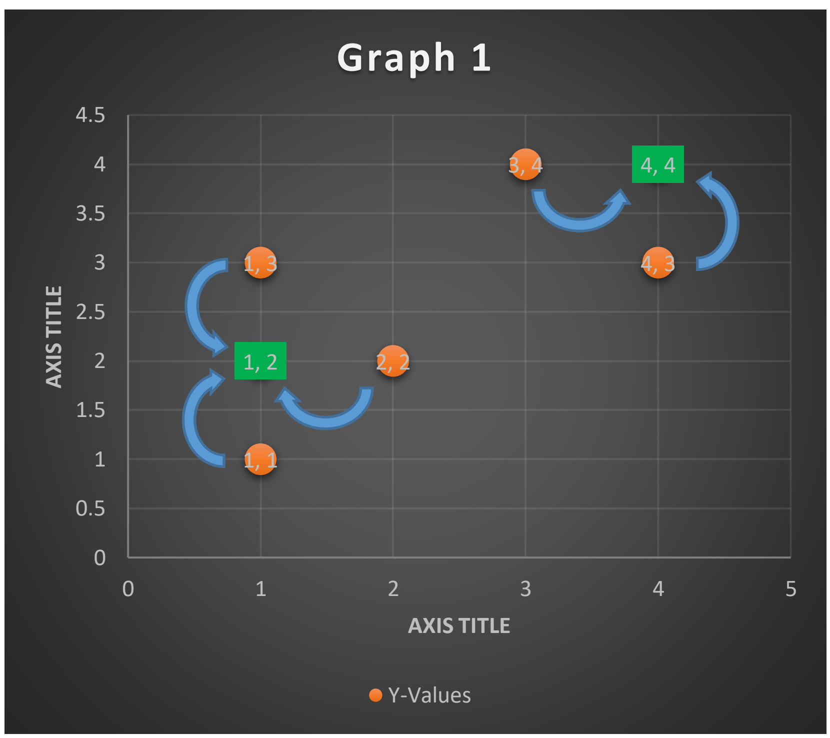
Solution: To explain this question I will take 7 nodes with the following positions.

Nodes ={ (1,1), (1,2), (1,3), (2,2), (3,4), (4,3),(4,4)}

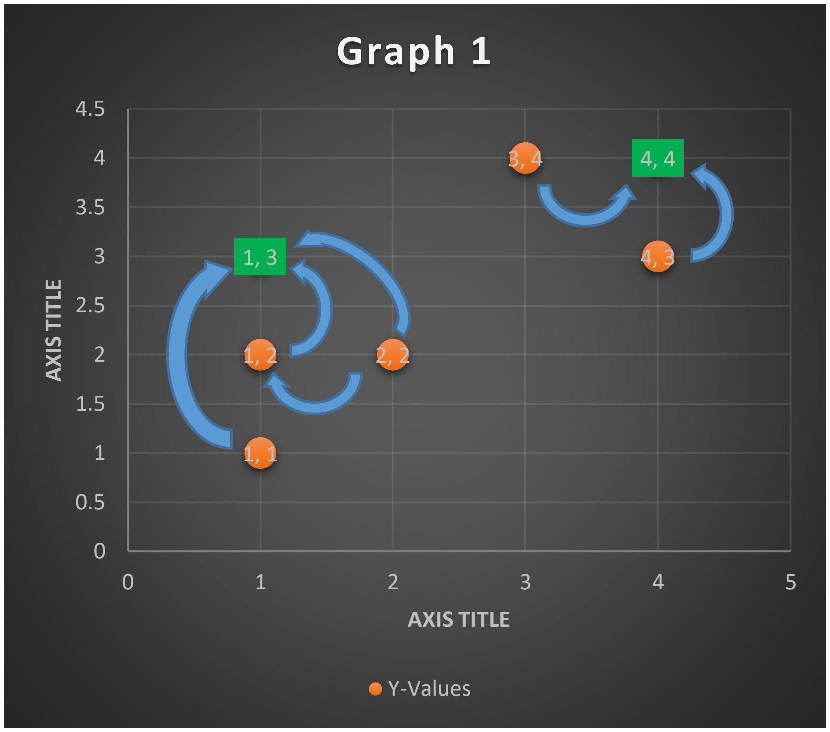
After plotting it on grid we get the following graph.



The optimal solution in this arrangement is r* = 1 for k=2 The following graph show it graphically. The Green Boxes represent centers. Every other node is at a distance of 1.



Let the greedy algorithm choses the third node (1,3). The next center will be (4,4) because its farthest from (1,3). As we are taking k=2 so it will stop finding other centers and start clustering. The center point (1,3) is near to {(1,1), (1,2), (2,2)} so they will form cluster. Center point (4,4) is near to {(3,4), (4,3)} so they will form cluster. The green squares represent the centers chosen by the greedy algorithm. Cluster 1 has radius = 2 and cluster 2 has radius=1 . The greedy algorithm choses the maximum one as r. so r=2 here



The algorithm choses two centers (1,3) and (4,4) and we have r=2 finally.

Now we have optimal solution and greedy search algorithm solution. By putting values of r and r* we get r/r*=2/1=2

Question 1: The question is solved through python . please see attached file. File Name= Connect_degree.py

(20%) Consider the following degree sequence: 3, 3, 3, 3, 1, 1, 1, 1, 1, 1. Is there a connected graph that can realize the degree sequence? If your answer is Yes, please give such a connected graph and explain how you obtain the graph.

Solution:

Step 1: total degree sum is even

Step 2. Check below output, it shows that the sequence of Nodes degree graphical.

Sequence of Nodes degree = [3, 3, 3, 3, 1, 1, 1, 1, 1, 1]
Sequence of Nodes degree = [2, 2, 2, 1, 1, 1, 1, 1, 1]
Sequence of Nodes degree = [1, 1, 1, 1, 1, 1, 1, 1]
Sequence of Nodes degree = [1, 1, 1, 1, 1, 1, 0]
Sequence of Nodes degree = [1, 1, 1, 1, 0, 0]
Sequence of Nodes degree = [1, 1, 0, 0, 0]
Sequence of Nodes degree = [0, 0, 0, 0]

Step 3. Sum of sequence of Nodes degree 18 >= 2(n-1)=18. So connected graph is possible.

Step 4. Constructing graph while adding lowest degree to highest degree node. Repeat the process until we have 0-degree sequence.

Degree sequence order = [1,1,1,1,1,1,3,3,3,3] → [0,0,0,0,1,1,2,2,2,2] → [0,0,0,0,0,0,2,2,1,1] → [0,0,0,0,0,0,1,1,0,0] → [0,0,0,0,0,0,0,0,0,0]

Now connecting in this order and we will get the following graph.